

Efficient, correct, unsupervised learning of context-sensitive languages

Alexander Clark

Department of Computer Science
Royal Holloway, University of London
`alexcl@cs.rhul.ac.uk`

CoNLL, July 2010

Classic CoNLL Problem

Unsupervised learning of syntax

Ultimate goal:

- Input: a large unannotated corpus
- Output: a grammar

Motivation

Understanding linguistics and language acquisition

- Not just crude constituent structure
- A generative grammar for the whole language – agreement, movement etc.

Non-standard methodology

Typical CoNLL method:

- 50-year old representation
- new ML techniques
- Heuristic approach
- Empirical evaluation
- Test on 1-3 languages
- Strong learning

Non-standard methodology

Typical CoNLL method:

- 50-year old representation
- new ML techniques
- Heuristic approach
- Empirical evaluation
- Test on 1-3 languages
- Strong learning

This paper:

- A new representation
- An old symbolic learning model
- Correct algorithm
- Proof
- A very large class of languages
- Weak learning

Two problems of grammar induction

Information theoretic problems

- Absence of negative data (Gold, 1967)
- VC-dimension (Vapnik, 1998)
- Sparsity, noise etc.

Many solutions: capacity control, regularisation, smoothing ...

Computational problems

Complexity of finding the best hypothesis

- Kearns and Valiant (1989), Abe and Warmuth (1992) ...
- Specific to certain classes of representation

In NLP these are largely ignored.

Some research strategies

Solve them both together

Too hard at the moment.

Only positive random samples

But we have unlimited computational power

- Horning (1969)
- Angluin (1988), Chater and Vitanyi (2007) ...

Only polynomial computation

But we have a good source of information:

- Positive examples
- Membership queries: we can ask whether $w \in L$

Goal: given some $L \subseteq \Sigma^*$: learn exactly which sentences are grammatical.

Distributional learning

Zellig Harris

Natural algorithmic idea:

- Look at the doggy
- Look at the car
- Look at the biscuit
- Look at the blue car
- the doggy is over there
- the biscuit is over there
- ...

Question: what classes of languages can be learned using this approach?

Distributional learning

Several reasons to take distributional learning seriously:

- Cognitively plausible (Saffran et al. 1996, Mintz, 2002)
- It works in practice: large scale lexical induction (Curran, J. 2003)
- Linguists use it as a constituent structure test (Carnie, A, 2008)
- Historically, PSGs were intended to be the output from distributional learning algorithms.

Chomsky (1968/2006)

“The concept of “phrase structure grammar” was explicitly designed to express the richest system that could reasonable be expected to result from the application of Harris-type procedures to a corpus.”

Distribution

Classic idea from structuralist linguistics:

Context (or *environment*)

A context is just a pair of strings $(l, r) \in \Sigma^* \times \Sigma^*$.

(l, r) combines with u to give lur

λ is the empty string; special context (λ, λ)

Given a language $L \subseteq \Sigma^*$:

Distribution of a string

$$C_L(u) = \{(l, r) \mid lur \in L\}$$

Clearly $(\lambda, \lambda) \in C_L(u)$ iff $u \in L$

“Distributional Learning” models/exploits the distribution of strings;





Observation table

K a set of strings and F a set of contexts

	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}
k_8										
k_7										
k_6										
k_5										
k_4										
k_3										
k_2										
k_1										

Observation table

K a set of strings and F a set of contexts

	(λ, λ)	(a, λ)	(λ, b)
λ			
a			
b			
ab			

Unrealistic assumption

We can fill in the table

$(aaabb, bccc)$ $(\lambda, abbccc)$ (aa, bbc) (abb, cc)
 (λ, λ) $(aaabbc, \lambda)$ $(aaab, bccc)$ $(aa, bbbc)$ $(abbb, cc)$

<i>bcc</i>									■
<i>aab</i>						■			
<i>bbcc</i>	■							■	
<i>bc</i>	■							■	
<i>abc</i>	■								
<i>aabb</i>	■					■			
<i>ab</i>	■					■			
<i>c</i>	■		■						■
<i>b</i>					■				
<i>a</i>	■			■			■		
λ	■	■				■		■	

Rearrange rows and columns

Some structure

(λ, λ) $(aaabb, bccc)$ $(\lambda, abbccc)$ $(aa, bbbc)$
 (aa, bbc) (abb, cc) $(aaabb, \lambda)$ $(aaab, bccc)$ $(abbb, cc)$

<i>bcc</i>									■
<i>aab</i>								■	
<i>abc</i>		■							
<i>aabb</i>	■	■							
<i>ab</i>	■	■							
λ	■	■	■	■					
<i>bbcc</i>		■	■						
<i>bc</i>		■	■						
<i>c</i>		■			■				■
<i>b</i>							■		
<i>a</i>		■				■		■	

Rectangles

Context free grammar

A non-terminal N

- $Y(N) = \{w \mid N \xRightarrow{*} w\}$
- $C(N) = \{(l, r) \mid S \xRightarrow{*} lNr\}$

Rectangle

If $(l, r) \in C(N)$ and $w \in Y(N)$, then $lwr \in L$
 N will be a rectangle in the observation table

Substitutable

$$L = \{a^n cb^n \mid n \geq 0\}$$

	(λ, λ)	(a, b)	(λ, cb)	(ac, b)	(a, λ)	(λ, b)	(aac, b)	(ac, λ)
$aacb$						■		
ac						■		
$acbb$				■				
cb				■				
$aacbb$	■	■						
acb	■	■						
c	■	■						
b							■	■
a			■					

Example

$$L = \{a^n b^n c^m \mid n, m \geq 0\} \cup \{a^m b^n c^n \mid n, m \geq 0\}$$

$$\begin{array}{cccccc} (aaabb, bccc) & (\lambda, abbccc) & (aa, bbc) & (abb, cc) & & \\ (\lambda, \lambda) & (aaabbc, \lambda) & (aaab, bccc) & (aa, bbbc) & (abbb, cc) & \end{array}$$

<i>bcc</i>								■
<i>aab</i>						■		
<i>bbcc</i>	■						■	
<i>bc</i>	■						■	
<i>abc</i>	■							
<i>aabb</i>	■				■			
<i>ab</i>	■				■			
<i>c</i>	■		■					■
<i>b</i>				■				
<i>a</i>	■		■			■		

Example

$$L = \{a^n b^n c^m \mid n, m \geq 0\} \cup \{a^m b^n c^n \mid n, m \geq 0\}$$

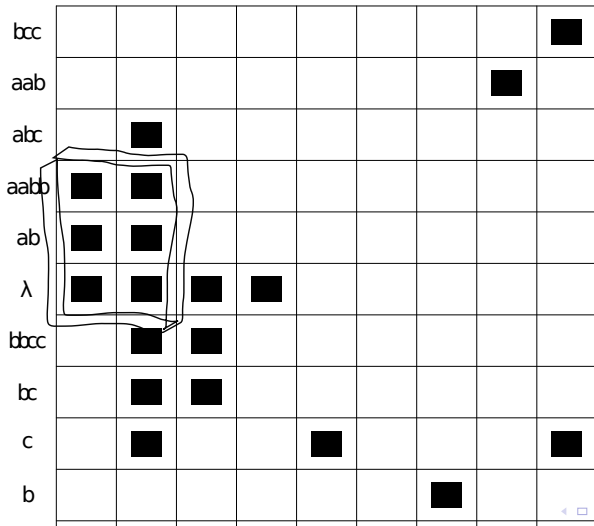
$$\begin{array}{cccc}
 (\lambda, \lambda) & (aaabb, bccc) & (\lambda, abbccc) & (aa, bbcc) \\
 (aa, bbc) & (abb, cc) & (aaabbc, \lambda) & (aaab, bccc) & (abbb, cc)
 \end{array}$$

<i>bcc</i>									■
<i>aab</i>								■	
<i>abc</i>		■							
<i>aabb</i>	■	■							
<i>ab</i>	■	■							
λ	■	■	■	■					
<i>bbcc</i>		■	■						
<i>bc</i>		■	■						
<i>c</i>		■			■				■
<i>b</i>							■		

Example

$$L = \{a^n b^n c^m \mid n, m \geq 0\} \cup \{a^m b^n c^n \mid n, m \geq 0\}$$

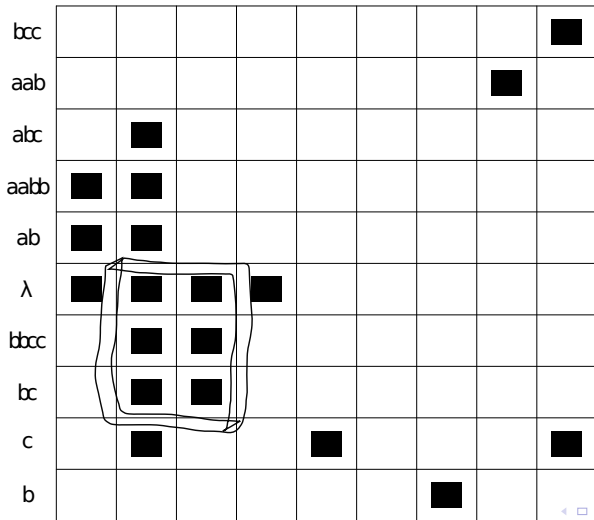
(λ, λ) $(aaabb, bccc)$ $(\lambda, abbccc)$ $(aa, bbcc)$
 $(aa, bbcc)$ (abb, cc) $(aaabbcc, \lambda)$ $(aaab, bccc)$ $(abbb, cc)$



Example

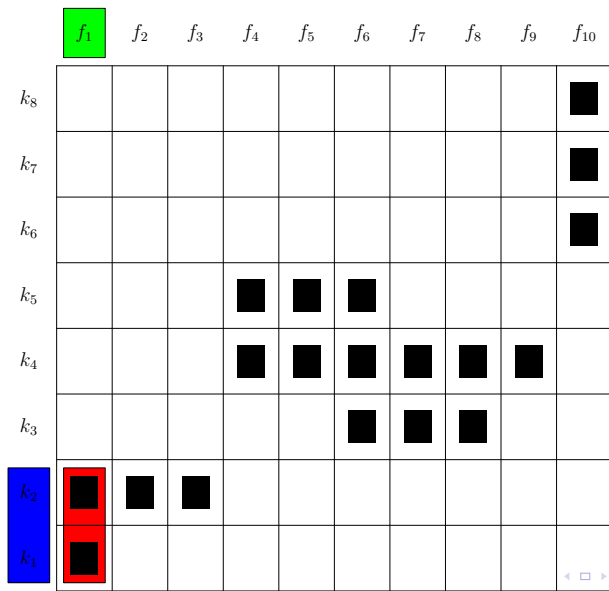
$$L = \{a^n b^n c^m \mid n, m \geq 0\} \cup \{a^m b^n c^n \mid n, m \geq 0\}$$

(λ, λ) $(aaabb, bccc)$ $(\lambda, abbccc)$ $(aa, bbcc)$
 $(aa, bbcc)$ (abb, cc) $(aaabbcc, \lambda)$ $(aaab, bccc)$ $(abbb, cc)$



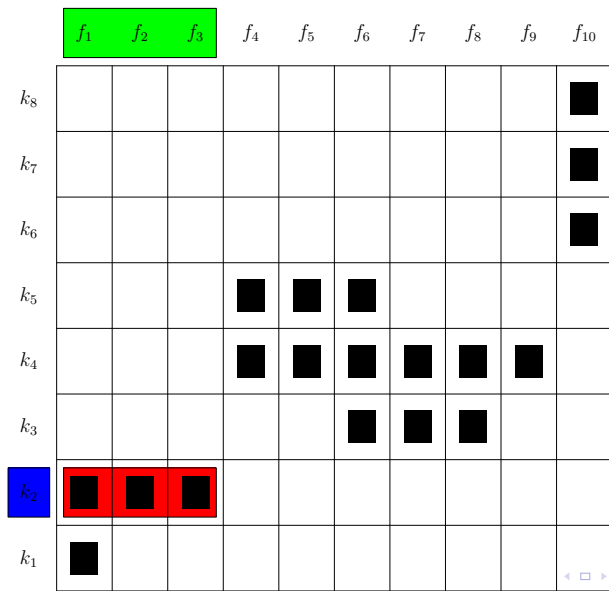
Concepts

Maximal rectangles



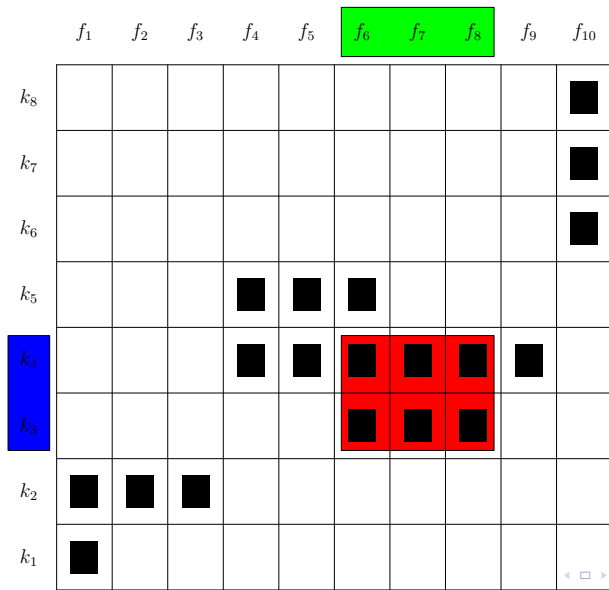
Concepts

Maximal rectangles



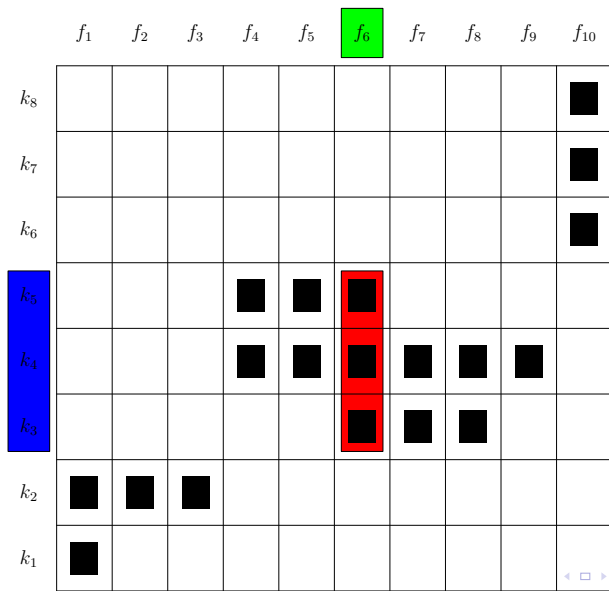
Concepts

Maximal rectangles



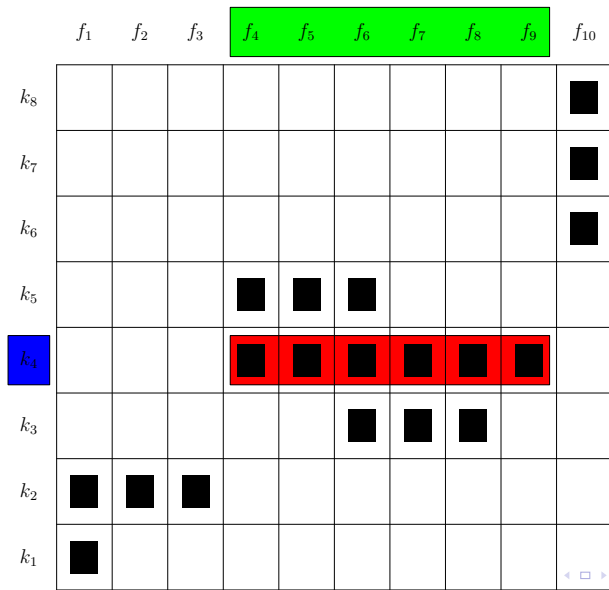
Concepts

Maximal rectangles



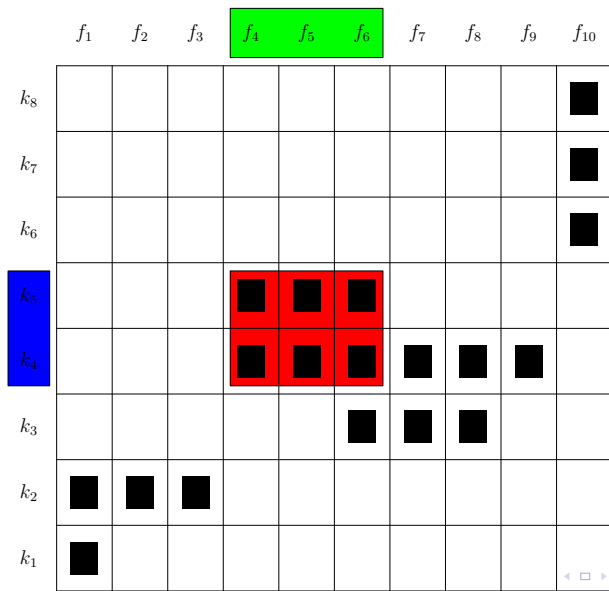
Concepts

Maximal rectangles



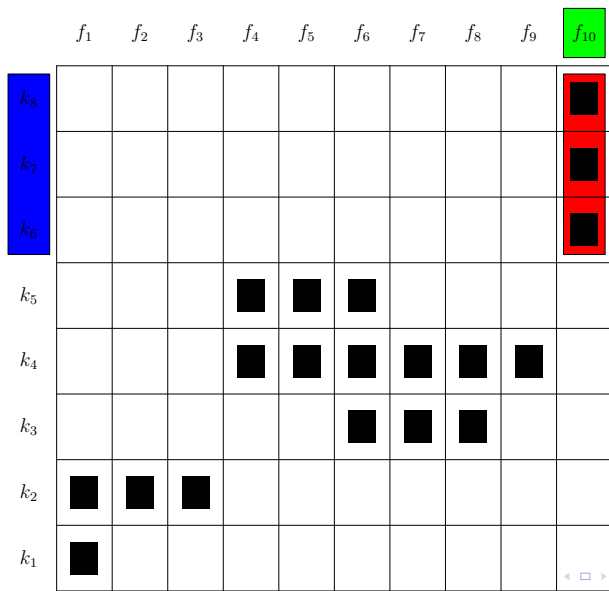
Concepts

Maximal rectangles

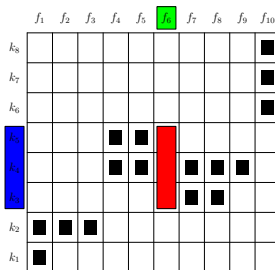
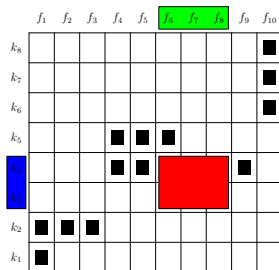


Concepts

Maximal rectangles



Partial order



Top and bottom

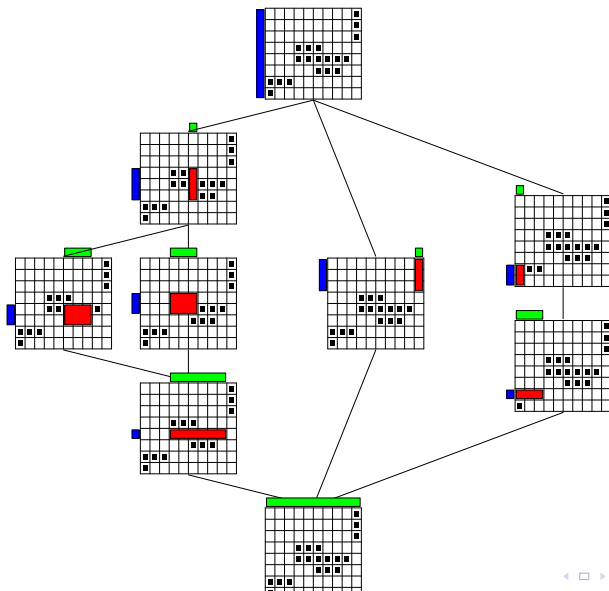
	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}
k_8										■
k_7										■
k_0										■
k_5				■	■	■				
k_4				■	■	■	■	■	■	
k_3						■	■	■		
k_2	■	■	■							
k_1	■									

Top and bottom

	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}
k_8										■
k_7										■
k_6										■
k_5				■	■	■				
k_4				■	■	■	■	■	■	
k_3						■	■	■		
k_2	■	■	■							
k_1	■									

Complete Lattice

Formal Concept Analysis

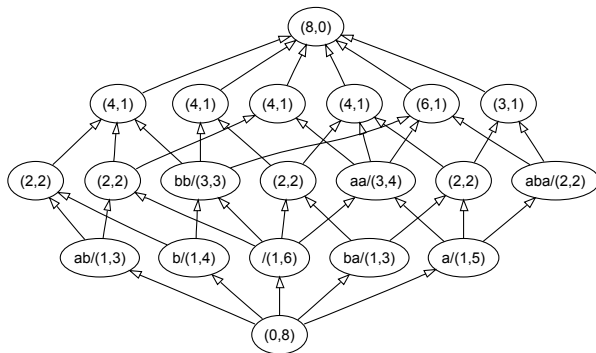


Lattice

Palindrome language over a, b

	(a, λ)	(aa, λ)	(ba, λ)	(λ, a)	(λ, λ)	(ab, λ)	(b, λ)	(λ, b)
aba	■		■					
bb	■				■		■	
ba		■	■				■	
ab					■	■		■
aa	■	■		■				■
b	■				■	■	■	
a	■	■	■	■				■

Lattice



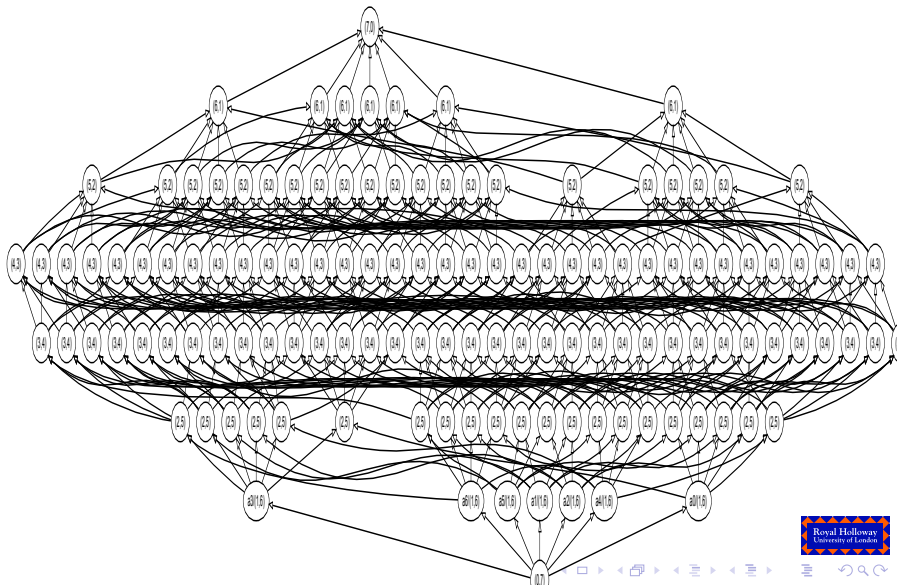
Lattice

Worst case can be exponential

	$(a1, \lambda)$	$(a3, \lambda)$	$(a5, \lambda)$			
	$(a0, \lambda)$	$(a2, \lambda)$	$(a4, \lambda)$	$(a6, \lambda)$		
$a6$	■	■	■	■	■	
$a5$	■	■	■	■		■
$a4$	■	■	■		■	■
$a3$	■	■		■	■	■
$a2$	■	■		■	■	■
$a1$	■		■	■	■	■
$a0$		■	■	■	■	■

Lattice

Cannot enumerate explicitly



Technical detail

Formal Concept Analysis

- These rectangles are “concepts” which form a complete lattice $\mathfrak{B}(K, L, F)$
- Each concept has a set of contexts: C . This defines a set of strings: $\{w \mid C_L(w) \supseteq C\}$.
- One concept consists of all strings in the language – defined by the context (λ, λ)
- We can define a greatest lower bound (*meet*) $X \wedge Y$.

Given the non-terminals, the rules are fixed

Suppose we have a grammar with non-terminals N, P, Q

- We have a rule $N \rightarrow PQ$
- This means that $Y(N) \supseteq Y(P)Y(Q)$.

Given the non-terminals, the rules are fixed

Suppose we have a grammar with non-terminals N, P, Q

- We have a rule $N \rightarrow PQ$
- This means that $Y(N) \supseteq Y(P)Y(Q)$.

Backwards

Given a collection of sets of strings X, Y, Z

Suppose $X \supseteq YZ$

Then we add a rule $X \rightarrow YZ$.

Concatenation

Concatenation

Define a concatenation operation $X \circ Y$

This needs a larger table: rows are KK , columns F .

Definition

Given two rectangles with strings $S_x, S_y \subseteq K$

- Concatenate them to get $S_x S_y \subseteq KK$
- Find the set of contexts shared by all these: C_Z
- Return the largest concept that contains C_Z

Distributional lattice grammars (DLGs)

Tuple $\langle K, D, F \rangle$

Recursive definition

$\phi : \Sigma^* \rightarrow \mathfrak{B}(K, D, F)$.

- for all $a \in \Sigma$
 $\phi(a) = \mathcal{C}(a)$
- for all w with $|w| > 1$,

$$\phi(w) = \bigwedge_{u,v \in \Sigma^+ : uv=w} \phi(u) \circ \phi(v)$$

Language defined:

Goal: $\phi(w)$ has the set of contexts C iff $C_L(w) \cap F = C$.

Define w to be in the language if $\phi(w)$ contains (λ, λ) .

Learnability

Search

Language is defined by choice of K and F
How can we find suitable K and F ?

Learnability

Search

Language is defined by choice of K and F
How can we find suitable K and F ?

Lemma 1

As we increase K the language defined by $\langle K, L, F \rangle$ decreases monotonically
It will always converge to a subset of L in a finite time

Learnability

Search

Language is defined by choice of K and F
How can we find suitable K and F ?

Lemma 1

As we increase K the language defined by $\langle K, L, F \rangle$ decreases monotonically
It will always converge to a subset of L in a finite time

Lemma 2

As we increase the set of contexts F the language monotonically increases.
Any sufficiently large set of contexts will do.

Search problem is trivial

Naive Algorithm

Start with $F = \{(\lambda, \lambda)\}$, $K = \Sigma \cup \{\lambda\}$

- If we see a string that is not in our hypothesis, the hypothesis is too small, and we add contexts to F
- Add strings to K without limit
- Fill in the table with MQs

Given that the data is adversarial we have to use an inefficient algorithm.

- Add all contexts of the data seen so far.
- We want it to stop changing hypotheses, so we test whether increasing K might change the hypothesis.

Power of Representation

Language class

Let \mathcal{L} be the set of all languages L such that there is a *finite* set of contexts F s.t. $L = L(\mathfrak{B}(\Sigma^*, L, F))$

Learnable class includes

- 1 All regular languages
- 2 Some but not all CFLs
- 3 Some non context free languages

Main result

Main Theorem

\mathcal{L} , represented by DLGs, can be identified in the limit from positive data and MQs

Polynomial update time

- Polynomial time bound is not quite sharp enough

Is the class large enough?

Wrong question

What CFGs are in the class?

Context free grammars

For a non-terminal N , can we find a finite set of contexts that pick out the strings generated by N ?

- $\{(l_1, r_1), \dots, (l_k, r_k)\}$
- If $w : l_1 w r_1, \dots, l_k w r_k \in L$, then $N \xRightarrow{*} w$

Any CFG with this property is in the learnable class.

Context sensitivity

- The lattice may have exponentially many elements in $|K| + |F|$
- (Exponentially many overlapping generalisations at different levels of generality)
- We cannot construct a CFG from all of them
- We can approximate parsing by taking the \wedge of all of the elements in one slot of the chart
- This means we have a CS representation but we maintain a cubic time parsing algorithm

We are forced to move to a CS representation to *solve* a computational problem.

Nice theory, but can I build a system with it?

A few obstacles to direct implementation:

- Queries can be substituted by probabilistic model
- Some problems with the partition function

Two changes

- Replace whole sentence context with a local window context
- Maybe: need a more refined learning argument
 - K could blow up
 - Number – case – gender explosion
 - Represent lattice more compactly through minimal elements

Conclusion

Jackendoff (2008)

- 1 Descriptive constraint: the class of languages must be sufficiently rich to represent natural languages
 - 2 Learnability constraint: there must be a way for the child to learn these representations from the data available
 - 3 Evolutionary constraint: it must not posit a rich, evolutionarily implausible language faculty
- Distributional Lattice Grammars potentially satisfy all three criteria

Summary

- A generalisation of distributional learning.
- By switching to a mildly CS representation we can efficiently consider every possible generalisation at the same time.
- Clean theoretical basis in the theory of residuated lattices.
- The first learning result for grammar induction that is **not obviously wrong**.
- There are no real alternatives to this and related results.

Any questions?

Not in DLG?

$$L = \{a^n b \mid n > 0\} \cup \{a^n c^m \mid m > n > 0\}$$

	(a, λ)	(aa, λ)	(aaa, λ)	(a^4, λ)	(a^5, λ)	(a^6, λ)	(a^7, λ)
c^9	■	■	■	■	■	■	■
c^8	■	■	■	■	■	■	■
c^7	■	■	■	■	■	■	■
c^6	■	■	■	■	■	■	■
c^5	■	■	■	■	■	■	
c^4	■	■	■	■	■		
ccc	■	■	■	■			
cc	■	■	■				
c	■	■					
b	■	■	■	■	■	■	■

Context sensitive example

Let $M = \{(a, b, c)^*\}$, we consider the language

$L = L_{abc} \cup L_{ab} \cup L_{ac}$ where $L_{ab} = \{wd | w \in M, |w|_a = |w|_b\}$,

$L_{ac} = \{we | w \in M, |w|_a = |w|_c\}$,

$L_{abc} = \{wf | w \in M, |w|_a = |w|_b = |w|_c\}$.

$F = \{(\lambda, \lambda), (\lambda, d), (\lambda, ad), (\lambda, bd), (\lambda, e), (\lambda, ae), (\lambda, ce), (\lambda, f), (ab, \lambda)\}$,

This is in the learnable class.